Neural Network Based Reinforcement Learning for Audio-Visual Gaze Control in Human-Robot Interaction

Stéphane Lathuilière[a,b], Benoit Massé[a,b], Pablo Mesejo[a,b], Radu Horaud[a,b]

[a] *Inria Grenoble Rhône-Alpes, Montbonnot-Saint-Martin, France*
[b] *Univ. Grenoble Alpes, Saint-Martin-d'Hères, France*

Abstract

This paper introduces a novel neural network-based reinforcement learning approach for robot gaze control. Our approach enables a robot to learn and to adapt its gaze control strategy for human-robot interaction neither with the use of external sensors nor with human supervision. The robot learns to focus its attention onto groups of people from its own audio-visual experiences, independently of the number of people, of their positions and of their physical appearances. In particular, we use a recurrent neural network architecture in combination with Q-learning to find an optimal action-selection policy; we pre-train the network using a simulated environment that mimics realistic scenarios that involve speaking/silent participants, thus avoiding the need of tedious sessions of a robot interacting with people. Our experimental evaluation suggests that the proposed method is robust in terms of parameter configuration, i.e. the selection of the parameter values employed by the method do not have a decisive impact on the performance. The best results are obtained when both audio and visual information is jointly used. Experiments with the Nao robot indicate that our framework is a step forward towards the autonomous learning of a socially acceptable gaze behavior.

1. Introduction

In recent years, there has been a growing interest in human-robot interaction (HRI), a research field dedicated to designing, evaluating and understanding robotic systems able to communicate with people [10]. The robotic agent must perceive humans and perform actions that, in turn, will have an impact on the interaction. For instance, it is known that the robot's verbal and gaze behavior has a strong effect on the turn-taking conduct of the participants [25]. Traditionally, HRI has been focused on the interaction between a single person with a robot. However, robots are increasingly part of groups and teams, *e.g.* performing delivery tasks in hospitals [16] or working closely alongside people on manufacturing floors [24]. In the case of the gaze control problem in a multi-person scenario, the fact of focusing on only one person would lead to omit important information and, therefore, to make wrong decisions. Indeed, the robot needs to follow a strategy to maximize useful information, and such a strategy is difficult to design for two main reasons. First, handling all the possible situations with a set of handcrafted rules would be laborious and most-likely sub-optimal, especially when combining several sensors. Second, the robot needs to be able to adapt its strategy to currently available data, as provided by its sensors, cameras and microphones in our case. For instance, if a companion robot enters a room with very bad acoustic conditions, the strategy needs to be adapted by decreasing the importance given to audio information.

In this paper, we consider the general problem of gaze control, with the specific goal of finding good policies to control the orientation of a robot head during informal group gatherings. In particular, we propose a methodology for a robotic system to be able to autonomously learn to focus its attention towards groups of people using audio-visual information. This is a very important topic of

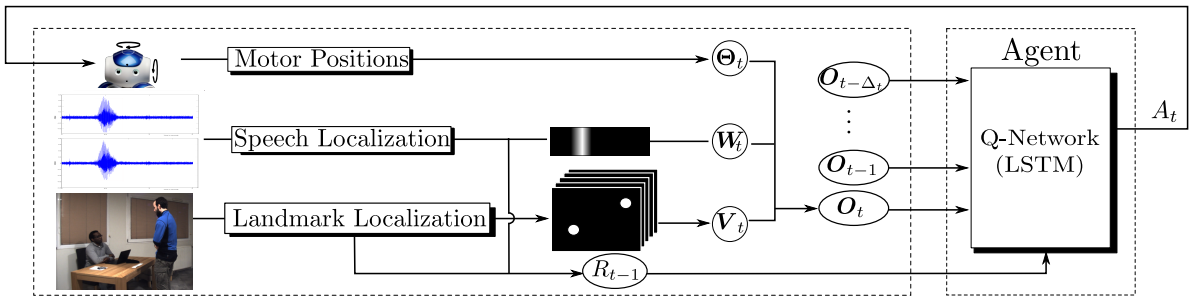


Figure 1: Overview of the proposed deep RL method for controlling the gaze of a robot. At each time index t, audio and visual data are represented as binary maps which, together with motor positions, form the set of observations $\boldsymbol{O}_t$. A motor action A_t (rotate the head left, right, up, down, or stay still) is selected based on past and present observations via maximization of current and future rewards. The rewards R are based on the number of visible persons as well as on the presence of speech sources in the camera field of view. We use a deep Q-network (DQN) model that can be learned both off-line and on-line. Please refer to Section 3 and Section 3.1 for the mathematical notations and detailed problem formulation.

research since perception requires not only making inferences from observations, but also making decisions about where to look next. More specifically, we want a robot to learn to find people in the environment, hence maximize the number of people present in its field of view, and favor people who speak. We believe this could be useful in many real scenarios, such as a conversation between a companion robot and a group of persons, where the robot needs to learn to look at people, in order to behave properly. The reason for using multiple sources of information can be found in recent HRI research suggesting that no single sensor can reliably serve robust interaction [20]. Importantly, when it comes to the employment of several sensing modalities in complex social interactions, it becomes difficult to implement an optimal policy based on handcrafted rules that take into consideration all possible situations that may occur. On the contrary, we propose to follow a data-driven approach to face such complexity. In particular, we propose to tackle this problem using a reinforcement learning (RL) approach [26]. RL is a machine learning paradigm in which agents learn by themselves by trial-and-error to achieve successful strategies. As opposed to supervised learning, there is no need for optimal decisions at training time, only a way to evaluate how good a decision is: a reward. This paradigm, inspired from behavioral psychology, may enable a robot to autonomously learn a policy that maximizes accumulated rewards. In our case, the agent, a robot companion, autonomously moves its head depending on its knowledge about the environment. This knowledge is called the *agent state*, and it is defined as a sequence of audio-visual observations, motor readings, actions, and rewards. In practice the optimal policy for making decisions is learned from the reward computed using detected faces of participants and sound sources being localized. The use of annotated data is not required to learn the best policy as the agent learns autonomously by trial-and-error in an unsupervised manner. Moreover, using our approach, it is not necessary to make any assumption about the number of people as well as their locations in the environment.

The use of RL techniques presents several advantages. First, training using optimal decisions is not required since the model learns from the reward obtained from previous decisions. The reward may well be viewed as a feedback signal that indicates how well the robot is doing at a given time step. Second, the robot must continuously make judgments so as to select good actions over bad ones. In this sense, the model can keep training at test time and hence it benefits from a higher adaptation ability. Finally, we avoid recourse to an annotated training set or calibration data. In our opinion, it seems entirely natural to use RL techniques to "educate" a robot, since recent neuro-scientific studies have suggested that reinforcement affects the way infants interact with their environment, including what they look at [1], and that gazing at faces is not innate, but that environmental importance influences the gazing behavior.

The contributions of this paper are the followings. First, robot gaze control is formulated as a reinforcement

learning problem, allowing the robot to autonomously learn its own gaze control strategy from multimodal data. Second, we use *deep* reinforcement learning to model the action-value function, and suggest several architectures based on LSTM (a recurrent neural network model) that allow us to experiment with both early- and late-fusion of audio and visual data. Third, we introduce a simulated environment that enables us to learn the proposed deep RL model without the need of spending hours of tedious interaction. Finally, by experimenting with both a publicly available dataset and with a real robot, we provide empirical evidence that our method achieves state-of-the-art performance.

2. Related Work

Reinforcement Learning has been successfully employed in different domains, including robotics [13]. The RL goal is to find a function, called a policy, which specifies which action to take in each state, so as to maximize some function (*e.g.*, the mean or expected discounted sum) of the sequence of rewards. Therefore, learning the suitable policy is the main challenge, and there are two main categories of methods to address it. First, policy-based methods define a space from the set of policies, and sample policies from this space. The reward is then used, together with optimization techniques, *e.g.* gradient-based methods, to increase the quality of subsequent sampled policies [30]. Second, value-based methods consist in estimating the expected reward for the set of possible actions, and the actual policy uses this value function to decide the suitable action, *e.g.* choose the action that maximizes the value-function. In particular, popular value-based methods include Q-learning [29] and its deep learning extension, Deep Q-Networks (or DQNs) [18].

There are several RL-based HRI methods relevant to our work. In [8] an RL algorithm is used for a robot to learn to play a game with a human partner. The algorithm uses vision and force/torque feedback to choose the motor commands. The uncertainty associated with human actions is modeled via a Gaussian process model, and Bayesian optimization selects an optimal action at each time step. In [17] RL is employed to adjust motion speed, timing, interaction distances, and gaze in the context of HRI. The reward is based on the amount of movement of the subject and the time spent gazing at the robot in one interaction. As external cameras are required, this cannot be easily applied in scenarios where the robot has to keep learning in a real environment. Moreover, the method is limited to the case of a single human participant. Another example of RL applied to HRI can be found in [27], where a human-provided reward is used to teach a robot. This idea of interactive RL is also exploited in [6] in the context of a table-cleaning robot. Visual and speech recognition are used to get advice from a parent-like trainer to enable the robot to learn a good policy efficiently. An extrinsic reward is used in [23] to learn how to point a camera towards the active speaker in a conversation. Audio information is used to determine where to point the camera, while the reward is provided using visual information: the active speaker raises a blue card that can be easily identified by the robot. The use of a multimodal deep Q-network (DQN) to learn human-like interactions is proposed in both [21] and [22]. The robot must choose an action to shake hands with a person. The reward is either negative, if the robot tries unsuccessfully to shake hands, positive, if the hand-shake is successful, or null otherwise. In practice, the reward is obtained from a sensor located in the hand of the robot and it takes fourteen training days to learn this skill successfully. To the best of our knowledge, the closest work to ours is [28] where an RL approach learns good policies to control the orientation of a mobile robot during social group conversations. The robot learns to turn its head towards the speaking person. However, their model is learned on simulated data that are restricted to a few predefined scenarios with static people and a predefined spatial organization of the groups.

Gaze control has been addressed in the framework of sensor-based servoing. In [4] an ad-hoc algorithm is proposed to detect, track, and involve multiple persons into an interaction, combining audio-visual observations. In a multi-person scenario, [3] investigated the complementary nature of tracking and visual servoing that enables the system to track several persons and to visually control the gaze such as to keep a selected person in the camera field of view. Also, in [31], a system for gaze control of socially interactive robots in multiple-person scenarios is presented. This method requires external sensors to locate

human participants.

However, in opposition to all these works, we aim at learning an optimal gaze control behavior using minimal supervision provided by a reward function, instead of adopting a handcrafted gaze control strategy. Importantly, our model requires neither external sensors nor human intervention to compute the reward, allowing the robot to autonomously learn where to gaze.

3. Reinforcement Learning for Gaze Control

We consider a robot task whose goal is to look at a group of people. Hence, the robot must learn by itself a gazing strategy via trials and errors. The desired robot action is to rotate its head (endowed with a camera and four microphones), such as to maximize the number of persons lying in the camera field-of-view. Moreover, the robot should prefer to look at speaking people instead of silent ones. The terms *agent* and *robot* will be used indistinctly.

Random variables and their realizations are denoted with uppercase and lowercase letters, respectively. Vectors and matrices are in bold italic. At each time index t, the agent gathers motor $\boldsymbol{\Theta}_t$, visual $\boldsymbol{V}_t$, and audio $\boldsymbol{W}_t$ observations and performs an action $A_t \in \mathcal{A}$ from an action set according to a policy π, i.e. controlling the two head motors such that the robot gazes in a selected direction. Once an action is performed, the agent receives a reward R_t, as explained in detail below.

Without loss of generality we consider the companion robot Nao whose head has two rotational degrees of freedom: motor readings correspond to pan and tilt angles, $\boldsymbol{\Theta}_t = (\Theta_t^1, \Theta_t^2)$. The values of these angles are relative to a reference head orientation, e.g. aligned with the robot body. This reference orientation together with the motor limits define the robot-centered *motor field-of-view* (M-FOV).

We use the multiple person detector of [5] to estimate two-dimensional visual landmarks, i.e. image coordinates, for each detected person, namely the nose, eyes, ears, neck, shoulders, elbows, wrists, hip, knees and ankles, or a total of $J = 18$ possible landmarks for each person. Based on the detection of these landmarks, one can determine the number of (totally or partially) observed persons, N_t, as well as the number of observed faces, F_t. Notice that in general the number of faces that are present in the image (i.e. detection of nose, eyes or ears) may be smaller than the number of detected persons. Since the camera is mounted onto the robot head, the landmarks are described in a head-centered reference system. Moreover, these landmarks are represented by J binary maps of size $K_v \times L_v$, namely $\boldsymbol{V}_t \in \{0,1\}^{K_v \times L_v \times J}$, where 1 (resp. zero) corresponds to the presence (resp. absence) of a landmark. Notice that this representation gathers all the detected landmarks associated with the N_t detected persons.

Audio observations are provided by the multi audio-source localization method described in [15]. Audio observations are also represented with a binary map of size $K_a \times L_a$, namely $\boldsymbol{W}_t \in \{0,1\}^{K_a \times L_a}$. A map cell is set to 1 if a speech source is detected at that cell and 0 otherwise. The audio map is robot-centered and hence it remains fixed whenever the robot turns its head. Moreover, the audio map spans an *acoustic field-of-view* (A-FOV), which is much wider than the *visual field-of-view* (V-FOV), associated with the camera mounted onto the head. The motor readings allow to estimate the relative alignment between the audio and visual maps and to determine whether a speech source lies within the visual field-of-view or not. This is represented by the binary variable $\Sigma_t \in \{0,1\}$, such that $\Sigma_t = 1$ if a speech source lies in the visual field-of-view and $\Sigma_t = 0$ if none of the speech sources lies inside the visual field-of-view.

Let $\boldsymbol{O}_t = \{\boldsymbol{\Theta}_t, \boldsymbol{V}_t, \boldsymbol{W}_t\}$ and let $\boldsymbol{S}_t = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_t\}$ denote the state variable. Let the set of actions be defined by $\mathcal{A} = \{\varnothing, \leftarrow, \uparrow, \rightarrow, \downarrow\}$, namely either remain in the same position or turn the head by a fixed angle in one of the four cardinal directions. We propose to define the reward R_t as follows:

$$R_t = F_{t+1} + \alpha \Sigma_{t+1}, \tag{1}$$

where $\alpha \geq 0$ is an adjustment parameter. Large α values return high rewards when speech sources lie within the camera field-of-view. We consider two types of rewards which are referred to in Section 4 as *Face_reward* ($\alpha = 0$) and *Speaker_reward* ($\alpha = 1$). Notice that the number of observed faces, F_t, is independent of the speaking state of each person. Upon the application at hand, the value of

α allows one to weight the importance given to speaking persons.

In RL, the model parameters are learned on sequences of states, actions and rewards, called episodes. At each time index t, an optimal action A_t should be chosen by maximizing the immediate and future rewards, $R_t, R_{t+1}, \ldots, R_T$. We make the standard assumption that future rewards are discounted by a factor γ that defines the importance of short-term rewards as opposed to long-term ones. We define the discounted future return $\bar{R}_t$ as the discounted sum of future rewards, $\bar{R}_t = \sum_{\tau=0}^{T-t} \gamma^\tau R_{\tau+t}$. If $\gamma = 0$, $\bar{R}_t = R_t$ and, consequently, we aim at maximizing only the immediate reward whereas when $\gamma \approx 1$, we favor policies that lead to better rewards in the long term. Considering a fixed value of γ, we now aim at maximizing $\bar{R}_t$ at each time index t. In other words, the goal is to learn a policy, $\pi(a_t, \boldsymbol{s}_t) = P(A_t = a_t | \boldsymbol{S}_t = \boldsymbol{s}_t)$ with $(a_t, \boldsymbol{s}_t) \in \mathcal{A} \times \mathcal{S}$, such that if the agent chooses its actions according to the policy π, the expected $\bar{R}_t$ should be maximized. The Q-function (or the action-value function) is defined as the expected future return from state $\boldsymbol{S}_t$, taking action A_t and then following any given policy π:

$$Q_\pi(\boldsymbol{s}_t, a_t) = \mathbb{E}_\pi[\bar{R}_t | \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t]. \tag{2}$$

Learning the best policy corresponds to the following optimization problem $Q^*(\boldsymbol{s}_t, a_t) = \max_\pi [Q_\pi(\boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t)]$. The optimal Q-function obeys the identity known as the Bellman equation:

$$Q^*(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{S}_{t+1}, R_t}\Big[R_t + \gamma \max_a (Q^*(\boldsymbol{S}_{t+1}, a)) \Big| \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t\Big] \tag{3}$$

This equation corresponds to the following intuition: if we have an estimator $Q^*(\boldsymbol{s}_t, a_t)$ for $\bar{R}_t$, the optimal action a_t is the one that leads to the largest expected $\bar{R}_t$. The recursive application of this policy leads to equation (3). A straightforward approach would consist in updating Q at each training step i with:

$$Q^i(\boldsymbol{s}_t, a_t) = \mathbb{E}_{\boldsymbol{S}_{t+1}, R_t}\Big[R_t + \gamma \max_a (Q^{i-1}(\boldsymbol{S}_{t+1}, a)) \Big| \boldsymbol{S}_t = \boldsymbol{s}_t, A_t = a_t\Big] \tag{4}$$

Following equation (4), we estimate each action-value $Q^i(\boldsymbol{s}_t, a_t)$ given that we follow, for the next time steps, the policy implied by Q^{i-1}. In practice, we approximate the true Q function by a function whose parameters must be learned. In our case, we employ a network $Q(\boldsymbol{s}, a, \boldsymbol{\omega})$ parametrized by weights $\boldsymbol{\omega}$ to estimate the Q-function $Q(\boldsymbol{s}, a, \boldsymbol{\omega}) \approx Q^*(\boldsymbol{s}, a)$. We minimize the following loss:

$$\mathcal{L}(\boldsymbol{\omega}_i) = \mathbb{E}_{\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{R}_t, \boldsymbol{S}_{t+1}}\Big[(Y_{i-1} - Q(\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{\omega}_i))^2\Big] \tag{5}$$

with $Y_{i-1} = R_t + \gamma \max_a (Q(\boldsymbol{S}_{t+1}, a, \boldsymbol{\omega}_{i-1}))$. This may be seen as minimizing the mean squared distance between approximations of the right- and left-hand sides of (4). In order to compute (5), we sample quadruplets $(\boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{R}_t, \boldsymbol{S}_{t+1})$ following the policy implied by Q^{i-1}:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(\boldsymbol{s}_t, a, \omega_{i-1}) \tag{6}$$

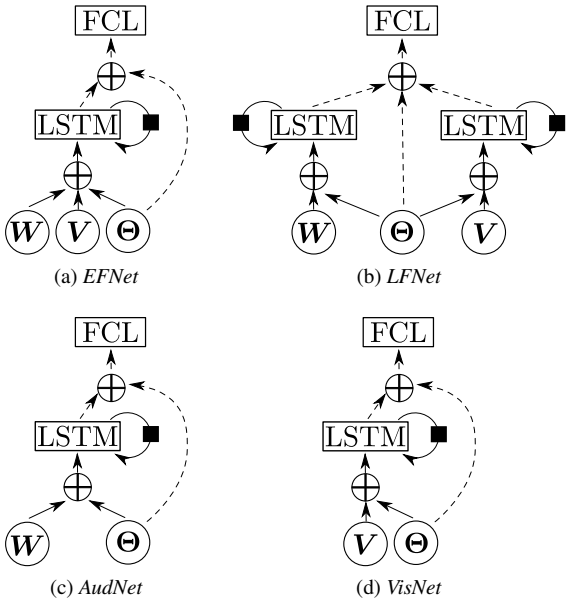
However, instead of sampling only according to (6), random actions a_t are taken in ϵ percents of the time steps in order to explore new strategies. This approach is known as epsilon-greedy policy. $\mathcal{L}$ is minimized over $\boldsymbol{\omega}_i$ by stochastic gradient descent. Refer to [19] for more technical details about the training algorithm.

3.1. Neural Network Architectures for Q-Learning

The Q-function is modeled by a neural network that takes as input part of the state variable $\boldsymbol{S}_t$, that we define as $\boldsymbol{S}_t^{\Delta t} = \{\boldsymbol{O}_{t-\Delta t} ... \boldsymbol{O}_t\}$. The output is a vector of size $\#\mathcal{A}$ that corresponds to each $Q_\pi(\boldsymbol{s}_t^{\Delta t}, a_t), a_t \in \mathcal{A}$, where $Q_\pi(\boldsymbol{s}_t^{\Delta t}, a_t)$ is built analogously to (2). Following [19], the output layer is a fully connected layer (FCL) with linear activations. We propose to use the long short-term memory (LSTM) [11] recurrent neural network to model the Q-function since recurrent neural networks are able to exhibit dynamic behavior for temporal sequences. LSTM are designed such as to prevent the back propagated errors from vanishing or exploding during training. We argue that LSTM is well suited for our task as it is capable of learning temporal dependencies better than other recurrent neural networks or than Markov models. In fact, our model needs to memorize the position and the motion of the people when it turns its head. When a person is not detected anymore, the network should be able to use previous detections back in time in order to predict the direction towards it should move. Batch normalization is

applied to the output of LSTM. The J channels of $\boldsymbol{V}_t$ are flattened before the LSTM layers.

Four different network architectures are described in this section and are evaluated in Section 4. In order to evaluate when the two streams of information (audio and video) need to be fused, we propose to compare two architectures: early fusion (*EFNet*) and late fusion (*LFNet*). In early fusion, the audio and visual features are combined into a single representation before modeling time dependencies, e.g. Figure 2a. Conversely, in late fusion, audio and visual features are modeled separately before fusing them, e.g. Figure 2b. In order to measure the impact of each modality, we also propose two more network architectures using either audio (*AudNet*) or vision (*VisNet*) information. Figure 2c displays *AudNet* and Figure 2d displays *VisNet*. Figure 2 employs a compact network representation where time is not explicitly shown, while Figure 3 depicts the unfolded representation of *EFNet* where each node is associated with one particular time instance. Both figures follow the graphical representation used in [9].



(a) *EFNet* (b) *LFNet* (c) *AudNet* (d) *VisNet*

Figure 2: Proposed architectures to model the Q-function. Dashed lines indicate connections only used in the last time step. Black squares represent a delay of a single time step. Encircled crosses depict the concatenation of inputs.

3.2. Pretraining on Simulated Environment

Training from scratch a DQN model can take a long time (in our case ∼150000 time steps to converge), and training directly on a robot would not be convenient for two reasons. First, it would entail a long period of training, since each physical action by the robot takes an amount of time that cannot be reduced neither by code optimization nor by increasing our computational capabilities. Second, in the case of HRI, participants would need to move in front of the robot for several hours or days (like in [21]). For these two reasons, we propose to use a transfer learning approach. The Q-function is first learned on a simulated environment, where we simulate people moving and talking, and it is then used to initialize the network employed by the robot. Importantly, the network learned from this simulated environment can be successfully used in the robot without the need of fine-tuning in real data. In this simulated environment, we do not need to generate images and sound signals, but only the observations and rewards the Q-Network receives as input.

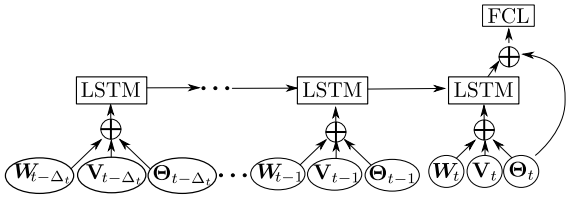


Figure 3: Unfolded representation of *EFNet* to better capture the sequential nature of the recurrent model. Encircled crosses depict the concatenation of inputs.

We consider that the robot can cover the field $[-1, 1]^2$ by moving its head, but can only visually observe the people within a small rectangular region $\mathcal{F}_t \subset [-1, 1]^2$ centered in position vector $\boldsymbol{\theta}_t$. The audio observations cover the whole reachable region $[-1, 1]^2$. On each episode, we simulate one or two persons moving with random speeds and accelerations within a field $[-\xi, \xi]^2$ where $\xi > 1$. In other words, people can go to regions that are unreachable for the robot. For each simulated person in the current episode, we consider the position and veloc-

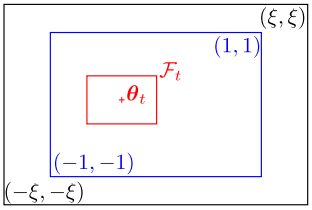


Figure 4: Diagram showing all fields used in the proposed simulated environment. The robot's field of view (in red) can move within the reachable field (in blue), whereas the participants can freely move within a larger field (in black).

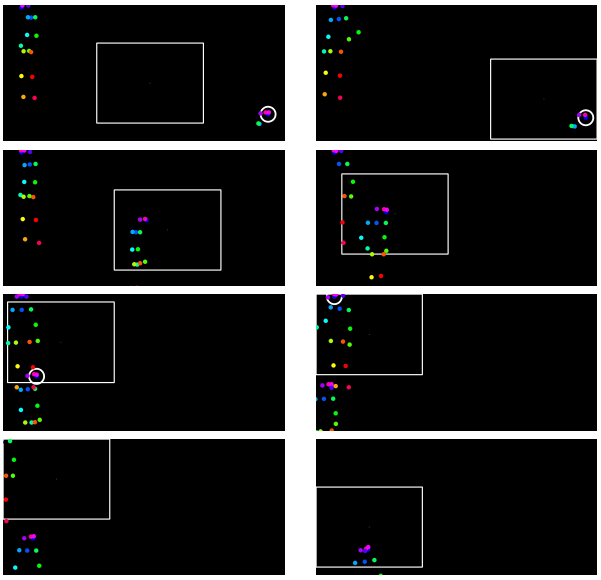


Figure 5: Illustrative sequence taken from the simulated environment and employed to pretrain our neural network-based RL approach. The moving square represents the *camera field-of-view* $\mathcal{F}_t$ of the robot. The colored circles represent the joints of a person in the environment. The large white circle represents a person speaking and, therefore, producing speech that can be detected by the speech localization system. Frames are displayed from top to bottom and left to right.

ity of their head at time t, $\boldsymbol{h}_t = (u_t^h, v_t^h) \in [-\xi, \xi]^2$ and $\dot{h} = (\dot{u}_t^h, \dot{v}_t^h) \in \mathbb{R}^2$, respectively. At each frame, the person can keep moving, stay without moving, or choose another random direction. The details of the simulated environment generator are given in Algorithm 1. In a real scenario, people can leave the scene so, in order to simulate this phenomenon, we consider two equally probable cases when a person is going out horizontally of the field ($v_t^h \notin [-\xi, \xi]$). In the first case, the person is deleted and instantly recreated on the other side of the field ($v_{t+1}^h = -v_t^h$) keeping the same velocity ($\dot{v}_{t+1}^h = \dot{v}_t^h$). In the second case, the person is going back towards the center ($v_{t+1}^h = v_t^h$ and ($\dot{v}_{t+1}^h = -\dot{v}_t^h$)). A similar approach is used when a person is going out vertically except that we do not create new persons on top of the field because that would imply the unrealistic sudden appearance of new legs within the field. Figure 4 displays a visual representation of the different fields (or areas) defined in our simulated environment, and Figure 5 shows an example of a sequence of frames taken from the simulated environment and used during training.

Moreover, in order to favor tracking abilities, we bias the person motion probabilities such that a person that is faraway from the robot head orientation has a low probability to move, and a person within *camera field-of-view* has a high probability to move. Thus, when there is nobody in the *camera field-of-view*, the robot cannot simply wait for a person to come in. On the contrary, the robot needs to track the persons that are visible. More precisely, we consider 4 different cases. First, when a person has never been seen by the robot, the person does not move. Second, when a person is in the robot field of view ($\boldsymbol{h}_t \in \mathcal{F}_t$), they move with a probability of 95%. Third, when the person is further than a threshold $\tau \in \mathbb{R}$ from the *camera field-of-view* ($\|\boldsymbol{h}_t - \boldsymbol{\Theta}_t\|_2 > \tau$), the probability of moving is only 25%. Finally, when the person is not visible but close to the *camera field-of-view* ($\|\boldsymbol{h}_t - \boldsymbol{\Theta}_t\|_2 < \tau$ and $\boldsymbol{h}_t \notin \mathcal{F}_t$), or when the person is unreachable ($\boldsymbol{h}_t \in [-\xi, \xi] \backslash [-1, 1]$), this probability is 85%. Regarding the simulation of missing detections, we randomly ignore some faces when computing the face features. Concerning the sound modality, we randomly choose between the following cases: 1 person speaking, 2 persons speaking, and nobody speaking. We use a Markov model to enforce continuity in the speaking status of the persons, and we also simulate wrong audio observations.

From, the head position, we need to generate the position of all body joints. To do so, we propose to collect a set $\mathcal{P}$ of poses from an external dataset (the AVDIAR dataset [7]). We use a multiple person pose estimator on this dataset and use the detected poses for our simulated environment. This task is not trivial since we need to simulate a realistic and consistent sequence of

poses. Applying tracking to the AVDIAR videos could provide good pose sequences, but we would suffer from three major drawbacks. First, we would have a tracking error that could affect the quality of the generated sequences. Second, each sequence would have a different and constant size, whereas we would like to simulate sequences without size constraints. Finally, the number of sequences would be relatively limited. In order to tackle these three concerns, we first standardize the output coordinates obtained on AVDIAR. Considering the pose p_t^n of the n^{th} person, we sample a subset $\mathcal{P}_t^M \subset \mathcal{P}$ of M poses. Then, we select the closest pose to the current pose: $p_{t+1}^n = \underset{p \in \Pi}{\operatorname{argmin}}\, d(p, p_t^n)$ where

$$d\left(\begin{pmatrix} u_1 \\ v_1 \\ s_1 \end{pmatrix}, \begin{pmatrix} u_2 \\ v_2 \\ s_2 \end{pmatrix}\right) = \frac{1}{\sum_{j=1}^{J} s_1^j s_2^j} \sum_{j=1}^{J} (s_1^j s_2^j) \sqrt{(u_1^j - u_2^j)^2 + (v_1^j - v_2^j)^2} \tag{7}$$

This distance is designed to face poses with different number of detected joints. It can be interpreted as an L_2 distance weighted by the number of visible joints in common. The intuition behind this sampling process is that when the size M of $\mathcal{P}_t^M$ increases, the probability of obtaining a pose closer to p_t^n increases. Consequently, the motion variability can be adjusted with the parameter M in order to obtain a natural motion. With this method we can obtain diverse sequences of any size.

Data: $\mathcal{P}$: a set of poses, δ: time-step
σ: velocity variance, M: pose continuity parameter

Randomly chose N in [1..3].
for $n \in [1..N]$ **do**
  Initialize
  $(\boldsymbol{h}_0^n, \dot{\boldsymbol{h}}_0^n) \sim \mathcal{U}([-1,1])^2 \times \mathcal{U}([-1,-0.5] \bigcup [0.5,1])^2$.
  Randomly chose p_0^n in $\mathcal{P}$.
end
for $t \in [1..T-1]$ **do**
  for $n \in [1..N]$ **do**
    Randomly chose $motion \in \{Stay, Move\}$
    if $motion = Move$ **then**
      if $\boldsymbol{h}_t^n \notin [-\xi, \xi]^2$ **then**
        The person is leaving the scene.
        See section 3.2.
      else
        $\boldsymbol{h}_{t+1}^n \leftarrow \boldsymbol{h}_t^n + \delta(\dot{\boldsymbol{h}}_t^n + \mathcal{N}((0,0), \sigma))$.
        $\dot{\boldsymbol{h}}_{t+1}^n \leftarrow \frac{1}{\delta}(\boldsymbol{h}_{t+1}^n - \boldsymbol{h}_t^n)$
      end
    else
      $\boldsymbol{h}_{t+1}^n \leftarrow \boldsymbol{h}_t^n$
      $\dot{\boldsymbol{h}}_{t+1}^n \sim \mathcal{U}([-1,-0.5] \bigcup [0.5,1])^2$
    end
    Draw $\mathcal{P}_t^M$, a random set of M elements of $\mathcal{P}$
    $p_{t+1}^n \leftarrow \underset{p \in \mathcal{P}_t^M}{\operatorname{argmin}}\, d(p, p_t^n)$
  end
end

Algorithm 1: Generation of simulated moving poses for our simulated environment.

4. Experiments

4.1. Evaluation with a Recorded Dataset

The evaluation of HRI systems is not an easy task. In order to fairly compare different models, we need to train and test the different models on the exact same data. In the context of RL and HRI, this is problematic because the data, i.e. what the robot actually sees and hears, depends on the action taken by the robot. Thus, we propose to first evaluate our model with the AVDIAR dataset [7]. This dataset was recorded with four microphones and one high-resolution camera (1920×1080 pixels). These images, due to their wide field of view, are suitable to simulate the motor field of view of the robot. In practical terms, only a small box of the full image simulates the robot's camera field of view. Concerning the observations, we employ visual and audio grids of sizes 7×5 in all our experiments with the AVDIAR dataset.

We employ 16 videos for training. The amount of training data is doubled by flipping the video and audio maps. In order to save computation time, the original videos are down-sampled to 1024×640 pixels. The size of the camera field of view where faces can be detected is set to 300×200 pixels using motion steps of 36 pixels each. These dimensions approximately correspond the coverage angle and motion of Nao. At the beginning of each episode, the position of the camera field of view is selected such that it contains no face. We noticed that this

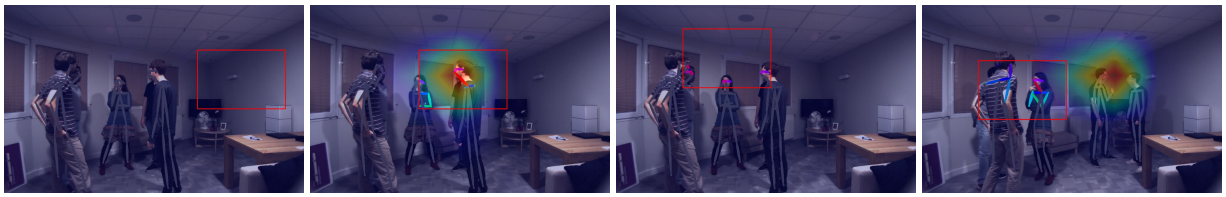


Figure 6: Example of a sequence from the AVDIAR dataset. The speech direction binary map is superimposed on the image, and the visible landmarks are displayed using a colored skeleton. The camera field of view (in red) is randomly initialized (far left), speech emitted by one of the persons is detected and hence the gaze is controlled (left). The agent is able to get all the persons in the field of view (right), and it gazes at a group of three persons while two other persons move apart (far right).

Figure 7: Example of a live sequence with two persons. First row shows an overview of the scene, including the participants and the robot. Second row shows the images gathered with the camera mounted onto the robot head. The robot head is first initialized in a position where no face is visible (first column), and the model uses the available landmarks (elbow and wrist) to find the person onto the right (second column). The robot detects the second person by looking around while keeping the first person in its field of view (third column), and gazes the two people walking together (fourth column).

initialization procedure favors the exploration abilities of the agent. To avoid a bias due to the initialization procedure, we used the same seed for all our experiments and iterated three times over the 10 test videos (20 when counting the flipped sequences). An action is taken every 5 frames (0.2 seconds).

Figure 6 shows a short sequence of the AVDIAR environment, displaying the whole field covered by the AVDIAR videos as well as the smaller field of view captured by the robot (the red rectangle in the figure). However, it is important to highlight that transferring the model learned using AVDIAR to Nao is problematic and did not work in our preliminary experiments. First, faces are almost always located at the same position (around the image center). Second, all videos are recorded indoors using only two different rooms, and participants are not moving too much. Finally, the audio setting is unrealistic for a robotics scenario, e.g. absence of motor noise. Therefore, the main reason for using the AVDIAR dataset is to compare our method with other methods in a generic setting.

4.2. Live Experiments with Nao

In order to carry out an online evaluation of our method, we performed experiments with a Nao robot. Nao has a 640×480 pixels cameras and four microphones. This robot is particularly well suited for HRI applications because of its design, hardware specifications and affordable cost. Nao's commercially available software can detect people, locate sounds, understand some spoken words,

synthesize speech and engage itself in simple and goal-directed dialogs. Our gaze control system is implemented on top of the NAOLab middleware [2] that synchronizes proprioceptive data (motor readings) and sensor information (image sequences and acoustic signals). The reason why we use a middleware is threefold. First, the implementation is platform-independent and, thus, easily portable. Platform-independence is crucial since we employ a transfer learning approach to transfer the model parameters, obtained with the proposed simulated environment, to the Nao software/hardware platform. Second, the use of external computational resources is transparent. This is also a crucial matter in our case, since visual processing is implemented on a GPU which is not available on-board of the robot. Third, the use of middleware makes prototyping much faster. For all these reasons, we employ the remote and modular layer-based middleware architecture named NAOLab. NAOLab consists of four layers: drivers, shared memory, synchronization engine and application programming interface (API). Each layer is divided into three modules devoted to vision, audio and proprioception, respectively. The last layer of NAOLab provides a general programming interface in C++ to handle the sensory data and to manage its actuators. NAOLab provides, at each time step, an image and the direction of the detected sound sources using [15, 14].

We now provide some implementation details specifically related to the Nao implementation. The delay between two successive observations is ∼0.3 seconds. The rotating head has a motor field-of-view of 180°. The head motion parameters are chosen such that a single action corresponds to 0.15 radians (∼9°) and 0.10 radians (∼6°) for horizontal and vertical motions, respectively. Concerning the observations, we employ a visual grid of size 7×5 and an audio grid of size 7×1 in all our experiments with Nao. Indeed, Nao has a planar microphone array and hence sound sources can only be located along the azimuth (horizontal) direction. Therefore the corresponding audio binary map is one-dimensional.

Figure 7 shows an example of a two-person scenario using the *LFNet* architecture. As shown in our recorded experiments [1] , we were able to transfer the exploration and tracking abilities learned using the simulated environment. Our model behaves well independently of the number of participants. The robot is first able to explore the space in order to find people. If only one person is found, the robot follows the person. If the person is static, the robot keeps the previously detected person in the field but keeps exploring the space locally aiming at finding more people. When more people appear, the robot tries to find a position that maximizes the number of people. The main failure cases are related to quick movements of the participants.

4.3. Implementation Details

By carefully selecting the resolution used to perform person detection along the method of [5], we were able to obtain visual landmarks in less than 100 ms. Considering that NAOLab gathers images at 10 FPS, this landmark estimator can be considered as fast enough for our purpose. Moreover, [5] follows a bottom-up approach, which allows us to speed-up landmark detection by skipping the costly association step.

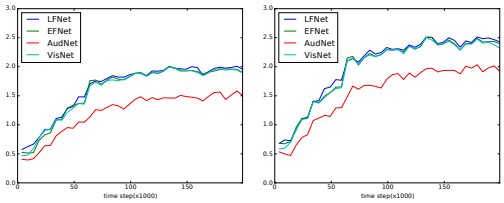
The parameters of our model are based on a preliminary experimentation. We set $\Delta_T = 4$ in all scenarios, such that each decision is based on the last 5 observations. The output size of LSTM is set to 30 (since a larger size does not provide an improvement in performance), and the output size of the FCL is set to 5 (one per action). We use a discount factor (γ) of 0.90. Concerning the training phases, we employed the Adam optimizer [12] and a batch size of 128. In order to help the model to explore the policy space, we use an ϵ-greedy algorithm: while training, a random action is chosen in ϵ% of the cases; we decrease linearly the ϵ value from $\epsilon = 90\%$ to $\epsilon = 10\%$ after 120000 iterations. The models were trained in approximately 45 minutes on both AVDIAR and the simulated environment. It is interesting to notice that we obtain this training time without using GPUs. A GPU is only needed for person detection and estimation of visual landmarks (in our case, a Nvidia GTX 1070 GPU).

In the simulated environment, the size of field in which the people can move is set to $\xi = 1.4$. In the case of

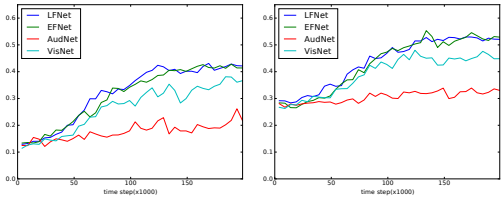
[1] A video showing offline and online experiments is available at `https://team.inria.fr/perception/research/deep-rl-for-gaze-control/`

Table 1: Comparison of the reward obtained with different architectures. The best results obtained are displayed in bold.

	AVDIAR				Simulated	
	Face		Speaker		*Face*	*Speaker*
Network	*Training*	*Test*	*Training*	*Test*		
AudNet	1.50 ± 0.03	1.47 ± 0.04	1.92 ± 0.02	1.82 ± 0.03	0.21 ± 0.01	0.33 ± 0.01
VisNet	1.89 ± 0.03	**1.85 ± 0.02**	2.32 ± 0.04	2.23 ± 0.03	0.37 ± 0.04	0.45 ± 0.06
EFNet	1.90 ± 0.03	1.81 ± 0.04	2.40 ± 0.02	2.22 ± 0.03	0.41 ± 0.03	**0.53 ± 0.03**
LFNet	**1.96 ± 0.02**	1.83 ± 0.02	**2.43 ± 0.02**	**2.29 ± 0.02**	**0.42 ± 0.01**	0.52 ± 0.03



(a) *Face_reward* on AVDIAR (b) *Speaker_reward* on AVDIAR



(c) *Face_reward* on Simulated (d) *Speaker_reward* on Simulated

Figure 8: Evolution of the reward obtained while training with the two proposed rewards on the AVDIAR dataset and on the simulated environment. We average over a 5000 time-step window for a cleaner visualization.

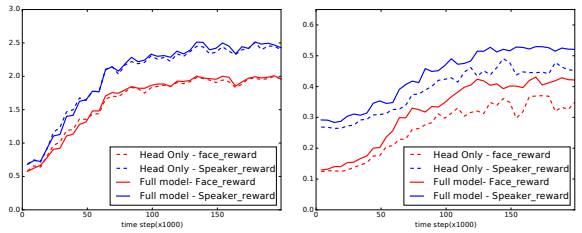
Nao, the audio observations are provided by the multiple speech-source localization method described in [15].

In all our experiments, we run five times each model and display the mean of five runs to lower the impact of the stochastic training procedure. On AVDIAR, the results on both training and test sets are reported in the tables. As described previously, the simulated environment is randomly generated in real time, so there is no need for a separated test set. Consequently, the mean reward over the last 10000 time steps is reported as test score.

4.4. Architecture Comparison

In Table 1, we compare the final reward obtained while training on the AVDIAR dataset and on our simulated environment with the two proposed rewards (*Face_reward* and *Speaker_reward*). Four different networks are tested: *EFNet*, *LFNet*, *VisNet*, and *AudNet*. The y-axis of Figure 8 shows the average reward per episode, with a clear growing trend as the training time passes (specially in the experiments with the AVDIAR dataset), meaning that the agent is learning (improving performance) from experience. On the the simulated environment, the best results are indistinctly provided by the late and early fusion strategies (*LFNet* and *EFNet*), showing that our model is able to effectively exploit the complementarity of both modalities. On the AVDIAR, the late fusion performs slightly better than the early fusion model. Globally, we observe that the rewards we obtain on AVDIAR are higher than those obtained on the simulated environment. We suggest two possible reasons. First, the simulated environment has been specifically designed to enforce exploration and tracking abilities. Consequently, it poses a more difficult problem to solve. Second, the number of people in AVDIAR is higher (about 4 in average), thus finding a first person to track would be easier. We notice that, on the AVDIAR dataset using the *Face_reward*, we obtain a mean reward greater than 1, meaning that, on average, our model can see more than one face per frame. We also observe that *AudNet* is the worst performing approach. However, it performs quite well on *AVDIAR* compared to the simulated environment. This behavior can be explained by the fact that, on *AVDIAR*, the speech source detector returns a 2D heatmap whereas only the angle is used in the simulated environment. As conclusion, we select *LFNet* to perform experiments on Nao.

Figure 9 displays the reward obtained when using only



(a) AVDIAR (b) Simulated

Figure 9: Evolution of the training reward obtained when using as visual observation the result of either the full-body pose estimation or the face location information.

faces as visual observation (dashed lines) in contrast to using the full-body pose estimation (continuous lines). We observe that on the simulated data, the rewards are significantly higher when using the full-body pose estimator. This figure intends to respond empirically to the legitimate question of why a full-body pose estimator is used instead of a simple face detector. From a qualitative point of view, the answer can be found in the type of situations that can solve one and the other. Let's imagine that the robot looks at the legs of a user; in case of using only a face detector, there is no clue that could help the robot to move up its head in order to see a face; however, if a human full-body pose detector is used, the detection of legs implies that there is a torso over them, and a head over the torso.

4.5. Parameter Study

In this section, we describe the experiments devoted to evaluate the impact of some of the principal parameters involved. More precisely, the impact of three parameters is analyzed. First, we compare different values for the discount factor γ that defines the importance of short-term rewards as opposed to long-term ones (see Section 3). Second, we compare different window sizes. It corresponds to the number of past observations that are used to make a decision (see Section 3.1). Finally, we compare different sizes for the LSTM network that is employed in all our proposed architectures (see Section 3.1). It corresponds to the dimension of the cell state and hidden state that are propagate by the LSTM.

Table 3: Comparison of the final reward obtained using different discounted factors (γ). The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold.

	AVDIAR		*Simulated*
γ	Training	Test	
25	**1.96 ± 0.02**	1.85 ± 0.02	0.33 ± 0.09
50	**1.96 ± 0.02**	**1.86 ± 0.03**	0.35 ± 0.08
75	**1.96 ± 0.02**	1.85 ± 0.02	**0.43 ± 0.11**
90	1.94 ± 0.02	1.83 ± 0.02	0.42 ± 0.12
99	1.95 ± 0.01	1.84 ± 0.02	0.42 ± 0.12

In Table 3, different discount factors are compared. With AVDIAR, high discount factors are prone to overfit as the difference in performance between training and test is large. With the simulated environment, low discount values perform worse because the agent needs to perform several actions to detect a face, as the environment is rather complex. Consequently, a model that is able to take into account future benefits of each action performs better. Finally, in Table 4, we compare different LSTM sizes. We observe that increasing the size doesn't lead to better results, which is an interesting outcome since, from a practical point of view, smaller LSTMs faster the training.

Different window sizes are compared in Table 2. We can conclude that the worst results are obtained when only the current observation is used (window size of 1). We also observe that, on AVDIAR, the model performs well even with short window lengths (2 and 3). In turn, with a more complex environment, as the proposed simulated environment, a longer window length tends to perform better. We interpret that using a larger window size helps the network to ignore the noisy observations and to remember the position of people that left the field of view. We report the training time for each window length. We observe that, using a smaller time window speeds up training since it avoids back-propagating the gradient deeply in the LSTM network.

4.6. Comparison with the State of the Art

We perform a comparative evaluation with the state of the art. To the best of our knowledge, no existing work addresses the problem of finding an optimal gaze policy in the HRI context. In [4] a heuristic that uses an

Table 2: Comparison of the final reward obtained using different window lengths (Δ_T). The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold. The training time is reported for each configuration.

	AVDIAR			*Simulated*	
$\Delta_T + 1$	Training	Test	Time(s$\times 10^3$)	Test	Time(s$\times 10^3$)
1	1.92 ± 0.03	1.82 ± 0.03	3.05 ± 0.22	0.26 ± 0.04	3.07 ± 0.15
2	1.94 ± 0.02	**1.85 ± 0.02**	2.25 ± 0.99	0.36 ± 0.04	3.09 ± 0.17
3	1.93 ± 0.01	1.84 ± 0.01	2.95 ± 0.38	0.42 ± 0.02	2.98 ± 0.27
5	1.94 ± 0.02	1.84 ± 0.02	3.30 ± 0.46	**0.43 ± 0.01**	3.40 ± 0.14
10	1.94 ± 0.02	1.84 ± 0.02	2.05 ± 0.22	0.40 ± 0.02	3.85 ± 0.36
20	**1.96 ± 0.01**	1.82 ± 0.02	3.00 ± 0.00	0.42 ± 0.02	5.35 ± 0.36
128	1.94 ± 0.02	1.82 ± 0.03	18.90 ± 0.77	0.41 ± 0.03	52.98 ± 5.23

Table 4: Comparison of the final reward obtained using different LSTM sizes. The mean and standard deviation over 5 runs are reported. The best average results obtained are displayed in bold.

LSTM size	AVDIAR		*Simulated*
	Training	Test	
30	**1.96 ± 0.01**	1.85 ± 0.03	0.42 ± 0.11
60	1.95 ± 0.02	1.86 ± 0.02	**0.43 ± 0.12**
120	1.92 ± 0.04	**1.87 ± 0.02**	0.41 ± 0.10

audio-visual input to detect, track and involve multiple interacting persons is proposed. Hence we compare our learned policy with their algorithm. On the simulated environment, as the speech source is only localized in the azimuthal plane (see section 4.3), we randomly gaze along the vertical axis in order to detect faces. In [3] two strategies are proposed to evaluate visually controlled head movements. A first strategy consists of following a person and rotating the robot head in order to align the person's face with the image center. A second strategy consists in randomly jumping every 3 seconds between persons. Obviously, the second strategy was designed as a toy experiment and does not correspond to a natural behavior. Therefore, we compare our RL approach with their first strategy. Unfortunately, the case where nobody is in the field of view is not considered in [3]. To be able to compare their method in the more general scenario addressed here, we propose the following handcrafted policy in the case no face is detected in the visual field of view: (i) *Rand*: A random action is chosen; (ii) *Center*: Go towards the center of the *acoustic field-of-view*; (iii) *Body*: If a limb is detected, the action ↑ is chosen in order to find the corresponding head, otherwise, *Rand* is followed, and (iv) *Audio*: Go towards the position of the last detected speaker.

Importantly, in our model the motor speed is limited, since the robot can only select unitary actions. When implementing other methods, one could argue that this speed limitation is inherent to our approach and that other methods may not suffer from it. However, it is not realistic to consider that the head can move between two opposite locations of the auditory field in two consecutive frames with an infinite speed. Therefore, we report two scores, first using the same speed value than the one used in our model (referred to as *equal*), and second by making the unrealistic assumption that the motor speed is infinite (referred to as *infinite*). This second evaluation protocol is therefore biased towards handcrafted methods. The results are reported in Table 5.

First, we notice that none of the handcrafted methods can compete with ours when considering the same motor speed. On both environments, *LFNet* largely outperforms all handcrafted models. This clearly justifies policy learning and the use of RL for gaze control. Concerning [3], *Center* obtains the best result among the [3]'s variances on AVDIAR and the worst on Simulated according to the *Face_reward* metric. This can be explained by the fact that, as mentioned in Section 4.1, most persons are located around the image center and, therefore, this dummy strategy works better than more sophisticated ones. A similar behavior can be observed with the *Speaker_reward* metric. We observe that in both environments using audio information, when no face is detected, improves the performance with respect to *Rand*.

Table 5: Comparison of the rewards obtained with different handcrafted policies. The performances of competitor methods are reported considering the two speed assumptions (*equal/infinite*) described in the text.

	AVDIAR		Simulated	
	Face_reward	*Speaker_reward*	*Face_reward*	*Speaker_reward*
Ban et al.[3]+*Rand*	1.19/1.21	1.45/1.59	0.25/0.26	0.40/0.37
Ban et al.[3]+*Center*	1.62/1.68	1.95/2.01	0.14/0.11	0.28/0.29
Ban et al.[3]+*Body*	1.23/1.20	1.40/1.52	0.27/0.26	0.39/0.37
Ban et al.[3]+*Audio*	1.54/1.63	1.84/2.06	0.32/0.39	0.43/0.48
Bennewitz et al.[4]	1.56/1.55	2.07/2.05	0.30/ **0.42**	0.35/0.50
LFNet	**1.83 ± 0.02**	**2.29 ± 0.02**	**0.42 ± 0.01**	**0.52 ± 0.03**

The second best performance on AVDIAR is obtained by [4] with *Speaker_reward*. On the simulated environment, [4] equals the score obtained by our proposal when making the unrealistic assumption of infinite motor speed. In that case, [4] is marginally inferior to our proposal according to the *Speaker_reward*. When considering equal speed limit, our RL approach significantly outperforms the handcrafted policy of [4] (26% and 48% higher according to *Face_reward* and *Speaker_reward*, respectively). All these results highlight the crucial importance of audio-visual fusion in the framework of RL and in the context of gaze control.

5. Conclusions

In this paper, we presented a neural network-based reinforcement learning approach to solve the gaze robot control problem. In particular, our agent is able to autonomously learn how to find people in the environment by maximizing the number of people present in its field of view while favoring people that speak. A simulated environment is used for pre-training prior to transfer learning to a real environment. Neither external sensors nor human intervention are necessary to compute the reward. Several architectures and rewards are compared on three different environments: two offline (real and simulated datasets) and real experiments using a robot. Our results suggest that combining audio and visual information leads to the best performance, as well as that pre-training on simulated data can even make unnecessary to train on real data. By thoroughly experimenting on a publicly available dataset and with a robot, we provide empirical evidence that our RL approach outperforms handcrafted strategies.

Acknowledgments

EU funding through the ERC Advanced Grant VHIA #340113 is greatly acknowledged.

References

[1] M. J. Arcaro, P. F. Schade, J. L. Vincent, C. R. Ponce, and M. S. Livingstone. Seeing faces is necessary for face-domain formation. *Nature Neuroscience*, 20(10):1404–1412, 2017.

[2] F. Badeig, Q. Pelorson, S. Arias, V. Drouard, I. Gebru, X. Li, G. Evangelidis, and R. Horaud. A distributed architecture for interacting with nao. In *ACM ICMI*, pages 385–386, 2015.

[3] Y. Ban, X. Alameda-Pineda, F. Badeig, S. Ba, and R. Horaud. Tracking a varying number of people with a visually-controlled robotic head. In *IEEE/RSJ IROS*, 2017.

[4] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, and S. Behnke. Towards a humanoid museum guide robot that interacts with multiple persons. In *IEEE-RAS*, pages 418–423, 2005.

[5] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *IEEE CVPR*, 2017.

[6] F. Cruz, G. I. Parisi, J. Twiefel, and S. Wermter. Multi-modal integration of dynamic audiovisual patterns for an interactive reinforcement learning scenario. In *IEEE/RSJ IROS*, pages 759–766, 2016.

[7] I. Gebru, S. Ba, X. Li, and R. Horaud. Audio-visual speaker diarization based on spatiotemporal bayesian fusion. *IEEE TPAMI*, 2017.

[8] A. Ghadirzadeh, J. Bütepage, A. Maki, D. Kragic, and M. Björkman. A sensorimotor reinforcement learning framework for physical Human-Robot Interaction. In *IEEE/RSJ IROS*, pages 2682–2688, 2016.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[10] M. A. Goodrich and A. C. Schultz. Human-robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2007.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[13] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *IJRR*, 2013.

[14] X. Li, L. Girin, F. Badeig, and R. Horaud. Reverberant sound localization with a robot head based on direct-path relative transfer function. In *IEEE/RSJ IROS*, 2016.

[15] X. Li, L. Girin, R. Horaud, and S. Gannot. Multiple-speaker localization based on direct-path features and likelihood maximization with spatial sparsity regularization. *IEEE/ACM TASLP*, 2017.

[16] S. Ljungblad, J. Kotrbova, M. Jacobsson, H. Cramer, and K. Niechwiadowicz. Hospital Robot at Work: Something Alien or an Intelligent Colleague? In *ACM CSCW*, pages 177–186, 2012.

[17] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita. Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning. *JRSJ*, 2006.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari With Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*. 2013.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[20] S. Pourmehr, J. Thomas, J. Bruce, J. Wawerla, and R. Vaughan. Robust sensor fusion for finding HRI partners in a crowd. In *IEEE ICRA*, pages 3272–3278, 2017.

[21] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro. Robot gains social intelligence through multimodal deep reinforcement learning. In *IEEE Humanoids*, pages 745–751, 2016.

[22] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro. Show, attend and interact: Perceivable human-robot social interaction through neural attention Q-network. In *IEEE ICRA*, pages 1639–1645, 2017.

[23] M. Rothbucher, C. Denk, and K. Diepold. Robotic gaze control using reinforcement learning. In *IEEE HAVE*, 2012.

[24] A. Sauppé and B. Mutlu. The Social Impact of a Robot Co-Worker in Industrial Settings. In *ACM CHI*, pages 3613–3622, 2015.

[25] G. Skantze, A. Hjalmarsson, and C. Oertel. Turn-taking, feedback and joint attention in situated human-robot interaction. *Speech Communication*, 65:50–66, 2014.

[26] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1st edition, 1998.

[27] A. L. Thomaz, G. Hoffman, and C. Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *IEEE RO-MAN*, pages 352–357, 2006.

[28] M. Vázquez, A. Steinfeld, and S. E. Hudson. Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach. In *IEEE RO-MAN*, 2016.

[29] C. J. C. H. Watkins and P. Dayan. Q-learning. *Mach. Learn.*, 1992.

[30] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn*, 1992.

[31] S.-S. Yun. A gaze control of socially interactive robots in multiple-person interaction. *Robotica*, 35(11):2122–2138, 2017.